IN THE UNITED STATES PATENT AND TRADEMARK OFFICE BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

On behalf of

Frederic **BOUTAUD**

APPELLANT

Application: **10/715,629** Examiner: **V. Lai**

Filed: **November 17, 2003** Group Art Unit: **2181**

Title: **DIGITAL SIGNAL PROCESSOR ARCHITECTURE WITH OPTIMIZED MEMORY ACCESS FOR CODE DISCONTINUITY**

APPELLANT'S BRIEF ON APPEAL

TABLE OF CONTENTS

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICANT:	Frederic **BOUTAUD**	**GROUP:**	2181
APPLICATION:	**10/715,629**	**EXAMINER:**	V. Lai
FILED:	November 17, 2003	**CONFIRMATION:**	4322

FOR: DIGITA AL PROCESSOR ARCHITECTURE WITH OPTIMIZED MEMORY ACCESS L SIGN FOR CODE DISCONTINUITY

Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313–1450

Sir:

APPEAL BRIEF FOR APPELLANTS

This Appeal Brief is being submitted in accordance with the Notification of Non-Compliant Appeal Brief, dated November 8, 2007 in connection with the above-identified application.

I. REAL PARTY OF INTEREST

The party of real interest to this appeal is the Assignee, Analog Devices, Incorporated.

II. RELATED APPEALS AND INTERFERENCES

The Appellant knows of no other pending appeals or interferences that are related to this instant appeal.

III. STATUS OF CLAIMS

Claims 1-14 have been previously presented in this application. Claims 1-5 have been allowed. Claims 6-14 are appealed.

IV. STATUS OF AMENDMENTS

The Appellant submitted a Response under 37 C.F.R. 1.116 on October 24, 2006, wherein the Appellant made no amendments to the claims. The Appellant filed a Request for a Pre-Appeal Brief Review on November 13, 2006. The Appellant received a Decision dated January 10, 2007. The Appellant filed an Appeal Brief January 16, 2007. The Examiner issued an Office Action dated June 12, 2007, indicating claims 1-5 have been allowed, but maintained the rejection of claims 6-14. The Appellant has not filed any other Responses and/or Amendments subsequent to the Office Action, dated June 12, 2007.

V. SUMMARY OF CLAIMED SUBJECT MATTER

In accordance with 37 C.F.R. 41.37(2)(c)(v), the following are concise explanations of the subject matter defined in the independent claims involved in this Appeal.

The Applicant has referenced both the originally filed specification page and line number and paragraph number. It is noted that particular reference numbers are not utilized because a particular reference number does necessarily pertain to the noted section of the claim, but the referenced Figure, as a whole, pertains to the noted section of the claim. **Moreover, it is respectfully submitted that the Figures (7, 11, 12, and 16) referenced below do not include reference numbers.**

A. Independent Claim 6

Independent claim 6 recites a method for accessing a unified memory in a micro-processing system having a microprocessor, a one level pipeline, and a two-phase clock, such that an instruction is executed in a single instruction cycle (see, for example, Figure 16; page 6, lines 18 and 19, of the originally filed specification; and paragraph **[0030]** of the published patent application). The method fetches a program instruction, corresponding to a second instruction cycle, from the unified memory during a first instruction cycle and determines if the fetched program instruction corresponding to the second instruction cycle is a conditional program code discontinuity (see, for example, Figure 7; page 11, line 27 through page 13, line 6 of the originally filed specification; and paragraphs **[0060]** through **[0065]** of the published patent application). The method also accesses the unified memory a first time during the second instruction cycle with a dummy access (see, for example, page 9, lines 3-10, of the originally filed specification paragraphs **[0045]** and **[0046]** of the published patent application) when it is determined that the fetched program instruction corresponding to the second instruction cycle is a conditional program code discontinuity (see, for example, Figure 7; page 11, line 27 through page 13, line 6 of the originally filed specification; and paragraphs **[0060]** through **[0065]** of the published patent application) and accesses the unified memory a

second time during the second instruction cycle to read a new instruction when it is determined the fetched program instruction corresponding to the second instruction cycle is a conditional program code discontinuity, thereby delaying instruction access from the unified memory for the second instruction cycle by a half cycle (see, for example, Figure 7; page 11, line 27 through page 13, line 6 of the originally filed specification; and paragraphs **[0060]** through **[0065]** of the published patent application).

B. Independent Claim 9

Independent claim 9 recites a method for accessing a unified memory in a micro-processing system having a microprocessor, a one level pipeline, and a two-phase clock, such that an instruction is executed in a single instruction cycle (see, for example, Figure 16; page 6, lines 18 and 19, of the originally filed specification; and paragraph **[0030]** of the published patent application). The method fetches a program instruction from the unified memory; determines if the fetched program instruction is a loop initiation instruction (see, for example, Figure 11; page 16, line 11 through page 17, line 12 of the originally filed specification; and paragraphs **[0078]** through **[0082]** of the published patent application); stores a first instruction of the loop in an instruction register when the fetched program instruction is a loop initiation instruction (see, for example, Figure 11; page 16, line 11 through page 17, line 12 of the originally filed specification; and paragraphs **[0078]** through **[0082]** of the published patent application); and executes the loop. The method also determines if a fetched instruction during the execution of the loop is a last instruction of the loop (see, for example, Figure 11; page 16, line 11 through page 17, line 12 of the originally filed specification; and paragraphs **[0078]** through **[0082]** of the published patent application); accesses the unified memory a first time, during the instruction cycle associated with the fetched last instruction of the loop, with a dummy access (see, for example, Figure 11; page 16, line 11 through page 17, line 12 of the originally filed specification; and paragraphs **[0078]** through **[0082]** of the published patent application); fetches the first instruction of the

loop from the instruction register, during the instruction cycle associated with the fetched last instruction of the loop (see, for example, Figure 11; page 16, line 11 through page 17, line 12 of the originally filed specification; and paragraphs **[0078]** through **[0082]** of the published patent application); and accesses the unified memory a second time, during the instruction cycle associated with the fetched last instruction of the loop, with a data access (see, for example, Figure 11; page 16, line 11 through page 17, line 12 of the originally filed specification; and paragraphs **[0078]** through **[0082]** of the published patent application).

C. Independent Claim 13

Independent claim 13 recites a method for accessing a unified memory in a micro-processing system during execution of a loop instruction. The method accesses a program instruction from the unified memory during a first instruction cycle; determines a type of program instruction; pre-fetches a second instruction from the unified memory; and saves the pre-fetched instruction in a register when it is determined that the type of program instruction is a first instruction of a loop (see, for example, Figure 12; page 17, line 13 through page 19, line 20 of the originally filed specification; and paragraphs **[0083]** through **[0093]** of the published patent application). The method also fetches an instruction from the register when it is determined that the type of program instruction is a last instruction of a loop; accesses the unified memory with a dummy access during execution of the last instruction of the loop; and accesses the unified memory, a second time, with a data access during execution of the last instruction of the loop (see, for example, Figure 12; page 17, line 13 through page 19, line 20 of the originally filed specification; and paragraphs **[0083]** through **[0093]** of the published patent application).

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

A. Rejection under 35 U.S.C. §102(b) over Morley

The grounds of rejection to be reviewed on appeal are whether claims 6-14 are patentable over Morley (US-A-4,276,594) in accordance with 35 U.S.C. §102(b).

VII. ARGUMENTS

A. Rejection of claims 6-14 under 35 U.S.C. §102(b) over Morley

Claims 6-14 have been rejected under 35 U.S.C. §102(b) as being anticipated by Morley (US-A-4,276,594). This rejection is respectfully traversed.

A. Independent Claim 6

With respect to independent claim 6, the Examiner alleges that Morley discloses fetching a program instruction from the unified memory during a first instruction cycle; determining if the fetched program instruction for a second instruction cycle is a conditional program code discontinuity; accessing the unified memory a first time during the second instruction cycle with a dummy access when it is determined that the program instruction accessed for a second instruction cycle is a conditional program code discontinuity; and accessing the unified memory a second time during the second instruction cycle to read a new instruction when it is determined the program instruction accessed for a second instruction cycle is a conditional program code discontinuity, thereby delaying the instruction access from the unified memory for the second instruction cycle by a half cycle. From the allegations, the Examiner concludes that the presently claimed invention of independent claim 6 is anticipated by the teachings of Morley.

As noted above, the Examiner alleges that Morley discloses accessing the unified memory a first time during the second instruction cycle with a dummy access when it is determined that the program instruction accessed for a second instruction

cycle is a conditional program code discontinuity and points to column 58, lines 17-20, of Morley to support the allegation. Column 58, lines 17-20, states:

> The 6800 accesses this RAM during the PHΦ of the MIO cycle. During serial I/O, this time is used to fetch/store characters and update pointers. For parallel I/O, this is used to fetch parameters for the PIA.

Contrary to the Examiner' position, this passage of Morley fails to discuss or disclose accessing the unified memory a first time during the second instruction cycle with a dummy access when it is determined that the program instruction accessed for a second instruction cycle is a conditional program code discontinuity, as set forth by independent claim 6.

Column 58, lines 17-20, of Morley merely teaches that the RAM may include a parallel I/O port. Column 58, lines 17-20, of Morley is void of any teachings directed to accessing the unified memory a first time during the second instruction cycle with a dummy access when it is determined that the program instruction accessed for a second instruction cycle is a conditional program code discontinuity, as set forth by independent claim 6.

The Examiner also alleges that Morley discloses accessing the unified memory a first time, during the instruction cycle associated with the fetched program instruction, with a dummy access.

The originally filed specification sets forth that "the cycle steal occurs at an instruction boundary by holding the next instruction from the program count and inserting a special "dummy" instruction into the pipeline," and " this dummy instruction does not change the program count nor starts a program instruction fetch."

On the other hand, contrary to the Examiner' position, Morley, at column 57 lines 9-25, teaches that although the RAM has three ports, only one port can be accessed at any time. Moreover, Morley teaches that four cycles are needed to transfer the 32-bit data. In other words, Morley fails to discuss or disclose any type of dummy access of the unified memory.

To refute this position by the Appellant, the Examiner asserts that page 8 of the originally filed specification teaches a dummy access is a special "dummy" instruction command inserted into the program code to perform an update. The portion of the originally filed specification being relied upon by the Examiner is a discussion as to how cycles from the pipeline can be stolen to move data to/from memory locations. In other words, as taught by the originally filed specification, the special "dummy" instruction "commands the core **11** to execute a data move transaction and perform associated register updates." This is an actual memory access, not a dummy memory access, because the purpose is to steal cycles from the pipeline to move data to/from memory locations.

On the other hand, as illustrated in Figure 7 and discussed in the supporting written description, when a program code discontinuity is encountered, a PM Memory dummy access is performed at the beginning of cycle **C** (upon the rising edge of the DSP core clock signal at the beginning of cycle **C**). However, the instruction **I2** is not fetched during this dummy memory access. The actual memory access is performed during cycle **C** (the PM instruction read access is triggered upon the falling edge of the DSP core clock signal). In this situation, a dummy memory access is used to steal time within an instruction cycle so that an instruction cycle does not need to be skipped due to program code discontinuity.

Thus, Morley fails to provide any basis for a finding of anticipation to the limitation corresponding to accessing the unified memory a first time, during the instruction cycle associated with the fetched program instruction, with a dummy memory access.

Lastly, as noted above, the Examiner alleges that Morley discloses accessing the unified memory a second time during the second instruction cycle to read a new instruction when it is determined the program instruction accessed for a second instruction cycle is a conditional program code discontinuity, thereby delaying the instruction access from the unified memory for the second instruction cycle by a half cycle and points to Figure 28 to support the allegation.

With respect to Figure 28, the read operation illustrates that during the PHΦ of the MIO cycle, the data from the memory is accessed and place on the bus; during the PH1 of the MIO cycle, the data sits on the bus; and during the PH2 of the MIO cycle, the data on the bus is uploaded or read. Thus, the memory is only access once per MIO cycle and fails to anticipate accessing the unified memory a second time, during the instruction cycle associated with the fetched program instruction, with a data access, as set forth by independent claim 6.

In summary, with respect to independent claim 6, Morley fails to teach (1) accessing the unified memory a first time during the second instruction cycle with a dummy access when it is determined that the program instruction accessed for a second instruction cycle is a conditional program code discontinuity; and/or (2) accessing the unified memory a second time, during the instruction cycle associated with the fetched program instruction, with a data access.

B. Independent Claim 9

With respect to independent claim 9, the Examiner alleges that Morley discloses fetching a program instruction from the unified memory; determining if the fetched program instruction is a loop initiation instruction; storing a first instruction of the loop in an instruction register when the fetched program instruction is a loop initiation instruction; executing the loop; determining if a fetched instruction during the execution of the loop is a last instruction of the loop; accessing the unified memory a first time, during the instruction cycle associated with the fetched last instruction of loop, with a dummy access; fetching the first instruction of the loop from the instruction register, during the instruction cycle associated with the fetched last instruction of loop; and accessing the unified memory a second time, during the instruction cycle associated with the fetched last instruction of loop, with a data access. From the allegations, the Examiner concludes that the presently claimed invention of independent claim 9 is anticipated by the teachings of Morley.

As noted above, the Examiner alleges that Morley discloses accessing the unified memory a first time, during the instruction cycle associated with the fetched last instruction of loop, with a dummy access and points to column 58, lines 17-20, of Morley to support the allegation. Column 58, lines 17-20, states:

> The 6800 accesses this RAM during the PHΦ of the MIO cycle. During serial I/O, this time is used to fetch/store characters and update pointers. For parallel I/O, this is used to fetch parameters for the PIA.

Contrary to the Examiner' position, this passage of Morley fails to discuss or disclose accessing the unified memory a first time, during the instruction cycle associated with the fetched last instruction of loop, with a dummy access, as set forth by independent claim 9.

Column 58, lines 17-20, of Morley merely teaches that the RAM may include a parallel I/O port. Column 58, lines 17-20, of Morley is void of any teachings directed to accessing the unified memory a first time, during the instruction cycle associated with the fetched last instruction of loop, with a dummy access, as set forth by independent claim 9.

The Examiner also alleges that Morley discloses accessing the unified memory a first time, during the instruction cycle associated with the fetched program instruction, with a dummy access.

The originally filed specification sets forth that "the cycle steal occurs at an instruction boundary by holding the next instruction from the program count and inserting a special "dummy" instruction into the pipeline," and " this dummy instruction does not change the program count nor starts a program instruction fetch."

On the other hand, contrary to the Examiner' position, Morley, at column 57 lines 9-25, teaches that although the RAM has three ports, only one port can be accessed at any time. Moreover, Morley teaches that four cycles are needed to transfer the 32-bit data. In other words, Morley fails to discuss or disclose any type of dummy access of the unified memory.

To refute this position by the Appellant, the Examiner asserts that page 8 of the originally filed specification teaches a dummy access is a special "dummy" instruction command inserted into the program code to perform an update. The portion of the originally filed specification being relied upon by the Examiner is a discussion as to how cycles from the pipeline can be stolen to move data to/from memory locations. In other words, as taught by the originally filed specification, the special "dummy" instruction "commands the core **11** to execute a data move transaction and perform associated register updates." This is an actual memory access, not a dummy memory access, because the purpose is to steal cycles from the pipeline to move data to/from memory locations.

On the other hand, as illustrated in Figure 7 and discussed in the supporting written description, when a program code discontinuity is encountered, a PM Memory dummy access is performed at the beginning of cycle **C** (upon the rising edge of the DSP core clock signal at the beginning of cycle **C**). However, the instruction **I2** is not fetched during this dummy memory access. The actual memory access is performed during cycle **C** (the PM instruction read access is triggered upon the falling edge of the DSP core clock signal). In this situation, a dummy memory access is used to steal time within an instruction cycle so that an instruction cycle does not need to be skipped due to program code discontinuity.

Thus, Morley fails to provide any basis for a finding of anticipation to the limitation corresponding to accessing the unified memory a first time, during the instruction cycle associated with the fetched program instruction, with a dummy access.

Moreover, as noted above, the Examiner alleges that Morley discloses accessing the unified memory a second time, during the instruction cycle associated with the fetched last instruction of loop, with a data access and points to Figure 28 to support the allegation.

With respect to Figure 28, the read operation illustrates that during the PHΦ of the MIO cycle, the data from the memory is accessed and place on the bus; during the PH1 of the MIO cycle, the data sits on the bus; and during the PH2 of the MIO cycle, the

data on the bus is uploaded or read. Thus, the memory is only access once per MIO cycle and fails to anticipate accessing the unified memory a second time, during the instruction cycle associated with the fetched program instruction, with a data access, as set forth by independent claim 9.

In summary, with respect to independent claim 9, Morley fails to teach (1) accessing the unified memory a first time, during the instruction cycle associated with the fetched last instruction of loop, with a dummy access; and/or (2) accessing the unified memory a second time, during the instruction cycle associated with the fetched program instruction, with a data access.

C. Dependent Claim 10

With respect to dependent claim 10, the Examiner points to Figure 28 and alleges that Morley discloses that a data access is a read data access.

Dependent claim 10 further limits claim 9 such that the claimed method accesses the unified memory a second time, during the instruction cycle associated with the fetched program instruction from the first access of the unified memory, with a read data access when it is determined that the fetched program instruction from the first access of the unified memory requires three unified memory accesses for proper execution of the fetched program instruction from the first access of the unified memory.

In contrast, Figure 28 of Morley merely illustrates the existence of read data accesses. Figure 28 of Morley fails to disclose or illustrate accessing the unified memory a second time, during the instruction cycle associated with the fetched program instruction from the first access of the unified memory, with a read data access when it is determined that the fetched program instruction from the first access of the unified memory requires three unified memory accesses for proper execution of the fetched program instruction from the first access of the unified memory, as established by dependent claim 10.

Therefore, Morley fails to anticipate the claimed invention of dependent claim 10.

D. Dependent Claim 11

With respect to dependent claim 11, the Examiner points to Figure 28 and alleges that Morley discloses that a data access is a write data access.

Dependent claim 11 further limits claim 9 such that the claimed method accesses the unified memory a second time, during the instruction cycle associated with the fetched program instruction from the first access of the unified memory, with a write data access when it is determined that the fetched program instruction from the first access of the unified memory requires three unified memory accesses for proper execution of the fetched program instruction from the first access of the unified memory.

In contrast, Figure 28 of Morley merely illustrates the existence of write data accesses. Figure 28 of Morley fails to disclose or illustrate accessing the unified memory a second time, during the instruction cycle associated with the fetched program instruction from the first access of the unified memory, with a write data access when it is determined that the fetched program instruction from the first access of the unified memory requires three unified memory accesses for proper execution of the fetched program instruction from the first access of the unified memory, as established by dependent claim 11.

Therefore, Morley fails to anticipate the claimed invention of dependent claim 11.

E. Independent Claim 13

With respect to independent claim 13, the Examiner alleges that Morley discloses accessing a program instruction from the unified memory during a first instruction cycle; determining a type of program instruction; pre-fetching a next instruction from the unified memory; saving the pre-fetched instruction in a register when it is determined that the type of program instruction is a first instruction of a loop; fetching a next instruction from the register when it is determined that the type of program instruction is a last instruction of a loop; accessing the unified memory with a dummy access during execution of the last instruction of the loop; and accessing the unified memory, a second time, with a data access during execution of the last instruction of the loop.

From the allegations, the Examiner concludes that the presently claimed invention of independent claim 13 is anticipated by the teachings of Morley.

As noted above, the Examiner alleges that Morley discloses accessing the unified memory with a dummy access during execution of the last instruction of the loop and points to column 58, lines 17-20, of Morley to support the allegation. Column 58, lines 17-20, states:

> The 6800 accesses this RAM during the PHΦ of the MIO cycle. During serial I/O, this time is used to fetch/store characters and update pointers. For parallel I/O, this is used to fetch parameters for the PIA.

Contrary to the Examiner' position, this passage of Morley fails to discuss or disclose accessing the unified memory with a dummy access during execution of the last instruction of the loop as set forth by independent claim 13.

Column 58, lines 17-20, of Morley merely teaches that the RAM may include a parallel I/O port. Column 58, lines 17-20, of Morley is void of any teachings directed to accessing the unified memory with a dummy access during execution of the last instruction of the loop, as set forth by independent claim 13.

The Examiner also alleges that Morley discloses accessing the unified memory a first time, during the instruction cycle associated with the fetched program instruction, with a dummy access.

The originally filed specification sets forth that "the cycle steal occurs at an instruction boundary by holding the next instruction from the program count and inserting a special "dummy" instruction into the pipeline," and " this dummy instruction does not change the program count nor starts a program instruction fetch."

On the other hand, contrary to the Examiner' position, Morley, at column 57 lines 9-25, teaches that although the RAM has three ports, only one port can be accessed at any time. Moreover, Morley teaches that four cycles are needed to transfer the 32-bit data. In other words, Morley fails to discuss or disclose any type of dummy access of the unified memory.

To refute this position by the Appellant, the Examiner asserts that page 8 of the originally filed specification teaches a dummy access is a special "dummy" instruction command inserted into the program code to perform an update. The portion of the originally filed specification being relied upon by the Examiner is a discussion as to how cycles from the pipeline can be stolen to move data to/from memory locations. In other words, as taught by the originally filed specification, the special "dummy" instruction "commands the core **11** to execute a data move transaction and perform associated register updates." This is an actual memory access, not a dummy memory access, because the purpose is to steal cycles from the pipeline to move data to/from memory locations.

On the other hand, as illustrated in Figure 7 and discussed in the supporting written description, when a program code discontinuity is encountered, a PM Memory dummy access is performed at the beginning of cycle **C** (upon the rising edge of the DSP core clock signal at the beginning of cycle **C**). However, the instruction **I2** is not fetched during this dummy memory access. The actual memory access is performed during cycle **C** (the PM instruction read access is triggered upon the falling edge of the DSP core clock signal). In this situation, a dummy memory access is used to steal time within an instruction cycle so that an instruction cycle does not need to be skipped due to program code discontinuity.

Thus, Morley fails to provide any basis for a finding of anticipation to the limitation corresponding to accessing the unified memory a first time, during the instruction cycle associated with the fetched program instruction, with a dummy access.

Moreover, as noted above, the Examiner alleges that Morley discloses accessing the unified memory, a second time, with a data access during execution of the last instruction of the loop and points to Figure 28 to support the allegation.

With respect to Figure 28, the read operation illustrates that during the PHΦ of the MIO cycle, the data from the memory is accessed and place on the bus; during the PH1 of the MIO cycle, the data sits on the bus; and during the PH2 of the MIO cycle, the data on the bus is uploaded or read. Thus, the memory is only access once per MIO

cycle and fails to anticipate accessing the unified memory, a second time, with a data access during execution of the last instruction of the loop, as set forth by independent claim 13.

In summary, with respect to independent claim 13, Morley fails to teach (1) accessing the unified memory with a dummy access during execution of the last instruction of the loop; and/or (2) accessing the unified memory, a second time, with a data access during execution of the last instruction of the loop.

Conclusion

Accordingly, for all the reasons set forth above, the Honorable Board is respectfully requested to reverse all the outstanding rejections. Also, an early indication of allowability is earnestly solicited.

Respectfully submitted,

Matthew E. Connors
Registration No. 33,298
Gauthier & Connors LLP
225 Franklin Street, Suite 2300
Boston, Massachusetts 02110
Telephone: (617) 426-9180
Extension 112

MEC/MJN/mjn

VIII. CLAIMS APPENDIX

6. A method for accessing a unified memory in a micro-processing system having a microprocessor, a one level pipeline, and a two-phase clock, such that an instruction is executed in a single instruction cycle, comprising:

(a) fetching a program instruction, corresponding to a second instruction cycle, from the unified memory during a first instruction cycle;

(b) determining if the fetched program instruction corresponding to the second instruction cycle is a conditional program code discontinuity;

(c) accessing the unified memory a first time during the second instruction cycle with a dummy access when it is determined that the fetched program instruction corresponding to the second instruction cycle is a conditional program code discontinuity; and

(d) accessing the unified memory a second time during the second instruction cycle to read a new instruction when it is determined the fetched program instruction corresponding to the second instruction cycle is a conditional program code discontinuity, thereby delaying instruction access from the unified memory for the second instruction cycle by a half cycle.

7. The method as claimed in claim 6, wherein the conditional program code discontinuity is a jump instruction.

8. The method as claimed in claim 6, wherein the conditional program code discontinuity is a call instruction.

9. A method for accessing a unified memory in a micro-processing system having a microprocessor, a one level pipeline, and a two-phase clock, such that an instruction is executed in a single instruction cycle, comprising:

(a) fetching a program instruction from the unified memory;

(b) determining if the fetched program instruction is a loop initiation instruction;

(c) storing a first instruction of the loop in an instruction register when the fetched program instruction is a loop initiation instruction;

(d) executing the loop;

(e) determining if a fetched instruction during the execution of the loop is a last instruction of the loop;

(f) accessing the unified memory a first time, during the instruction cycle associated with the fetched last instruction of the loop, with a dummy access;

(g) fetching the first instruction of the loop from the instruction register, during the instruction cycle associated with the fetched last instruction of the loop; and

(h) accessing the unified memory a second time, during the instruction cycle associated with the fetched last instruction of the loop, with a data access.

10. The method as claimed in claim 9, wherein the data access is a read data access.

11. The method as claimed in claim 9, wherein the data access is a write data access.

12. The method as claimed in claim 9, wherein the instruction register is an instruction stack, thereby enabling program instruction fetches for nested loops.

13. A method for accessing a unified memory in a micro-processing system during execution of a loop instruction, comprising:

(a) accessing a program instruction from the unified memory during a first instruction cycle;

(b) determining a type of program instruction;

(c) pre-fetching a second instruction from the unified memory;

(d) saving the pre-fetched instruction in a register when it is determined that the type of program instruction is a first instruction of a loop;

(e) fetching an instruction from the register when it is determined that the type of program instruction is a last instruction of a loop;

(f) accessing the unified memory with a dummy access during execution of the last instruction of the loop; and

(g) accessing the unified memory, a second time, with a data access during execution of the last instruction of the loop.

14. The method as claimed in claim 13, wherein the pre-fetched instruction is saved in a stack when it is determined that the type of program instruction is first instruction of a loop to enable nested loops and interruptible loops, and a next instruction is fetched from the stack when it is determined that the type of program instruction is a last instruction of the loop.

IX. EVIDENCE APPENDIX

NONE

X. RELATED PROCEEDINGS APPENDIX

NONE